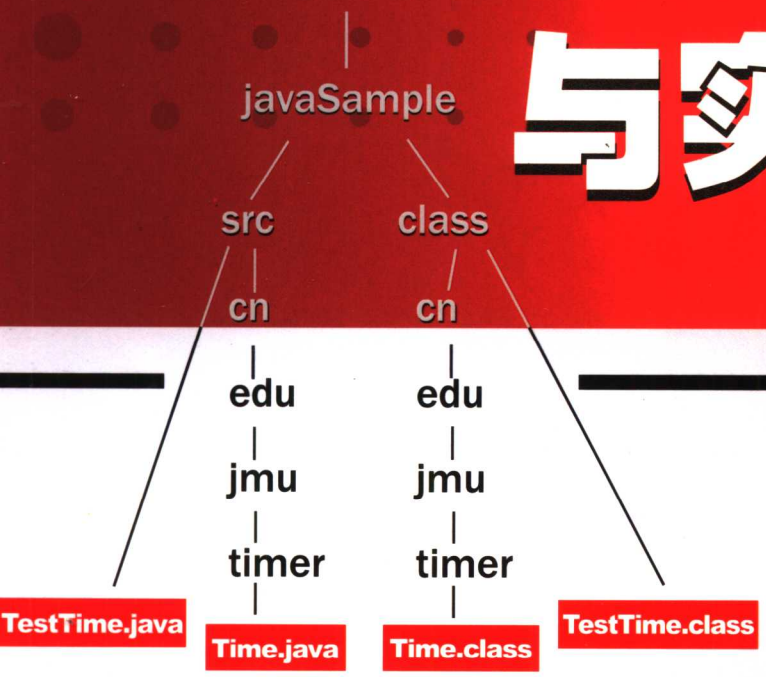
Java 2

面向对象程序设计基础与实例解析



陈艳华 主编

清华大学出版社

Java 2 面向对象程序设计基础与实例解析

陈艳华　主编

清华大学出版社
北　京

内 容 简 介

本书是作者根据最新计算机教学大纲，并总结多年从事 Java 语言程序设计的教学经验编写而成的。

本书全面讲解了 Java 的基础内容和编程方法，在内容的深度和广度方面都给予了认真的考虑，在类、对象、继承、接口等重要的基础知识上侧重深度，而在实用类的讲解上侧重广度。另外，还以具体的案例介绍了本书知识的综合应用；而且每章都配有一定数量的习题或思考题，便于读者复习参考。通过学习，读者可以掌握 Java 面向对象编程的思想和 Java 编程的技术。

本书的特点是知识内容循序渐进，通俗易懂，概念清晰，思路新颖；适合作为各类院校的相关课程教材，也可作为计算机爱好者学习面向对象程序设计的自学教材。

图书在版编目(CIP)数据

Java 2 面向对象程序设计基础与实例解析/陈艳华主编. —北京：清华大学出版社，2007.5
ISBN 978-7-302-15009-1

Ⅰ.J…　Ⅱ. 陈…　Ⅲ. JAVA 语言—程序设计　Ⅳ. TP312

中国版本图书馆 CIP 数据核字(2007)第 048341 号

责任编辑：邹　杰　宋延清
封面设计：杨玉兰
版式设计：北京东方人华科技有限公司
责任校对：周剑云
责任印制：何　芊
出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　**邮　　编**：100084
c-service@tup.tsinghua.edu.cn
社 总 机：010-62770175　**邮购热线**：010-62786544
投稿咨询：010-62772015　**客户服务**：010-62776969
印 装 者：北京宏伟双华印刷有限公司
经　　销：全国新华书店
开　　本：185×260　**印张**：23.75　**字数**：573 千字
版　　次：2007 年 5 月第 1 版　　**印　　次**：2007 年 5 月第1 次印刷
印　　数：1～4000
定　　价：34.00 元

前　言

随着计算机技术和网络技术的发展，Java 语言作为面向对象且跨平台的编程语言，自 1996 年正式发布以来，迅速成为 IT 领域里的主流编程语言。面向对象的 Java 语言具备一次编程、任何地方均可运行的能力，使其成为软件服务提供商和系统集成商用以支持多种操作系统和硬件平台的首选解决方案。Java 作为软件开发的一种革命性的技术，其地位已确定。Java 技术已是当今世界信息技术的主流之一。

本书共有 13 章。第 1 章介绍 Java 语言的历史发展和特点、Java 虚拟机和 Java 平台。第 2 章介绍 Java 的两个编辑器 JDK 和 JCreator，及其配置。第 3 章主要介绍 Java 语言的基础，包括 Java 语言标识符、关键字、数据类型、字符串类、表达式、运算符、数据类型的转换和 Java 数组，包括一般功能语句、控制流语句和异常处理语句。第 4 章介绍面向对象的 Java 编程方法，包括定义 Java 类、创建和使用 Java 对象等。第 5 章介绍 Java 中的包、异常处理机制及接口技术，包括建立 Java 包、布局源文件和相关参数 classpath 的设置、异常的基本知识。第 6 章介绍线程操作，包括线程创建、调度、优先级、同步和线程通信等。第 7 章介绍 Java 中的输入/输出，流如何定义，Java 如何实现基本的输入输出。第 8 章介绍学习与使用 Java 提供的网络接口来编写网络应用程序。第 9 章介绍创建图形用户界面的方法。第 10 章介绍 Java Applet 程序设计。第 11 章介绍 JDBC 的体系结构及其重要接口，使用 JDBC 连接和访问数据库的方法。第 12 章介绍 JSP 与 Servlet 的联系与区别。

本书的特点是知识内容的阐述循序渐进，通俗易懂，概念清晰，思路新颖，而且每章都配有一定数量的习题或思考题，便于读者复习参考。

本书集成了各位编者多年的 Java 教学和实践经验的精华。书中实例均为完整可执行的 Java 程序，并附有运行结果。书中图表、图形具有良好的综述和示意能力，方便读者查阅和记忆。

限于作者水平，加之本书覆盖面广，其中若有不妥之处，敬请广大读者批评指正。

编　者

目　录

第 1 章　Java 入门

Java 语言是目前推广速度最快的程序设计语言，它采用面向对象的编程技术，功能强大又简单易学。Java 伴随着 Internet 的发展而成熟，内置了多线程和网络支持能力，可以说是网络世界的通用语言。本章将介绍 Java 语言的基本特点和开发的一般过程。

1.1　Java 概述

本节介绍 Java 的发展、Java 语言的特点、与 C/C++的区别以及 Java 的应用。

1. Java 的发展

1991 年初，美国的 Sun Microsystems 公司投资了一个名为 Green 的研究项目，负责研究消费性电子产品及相关软件的开发。研究小组以 C/C++语言为蓝本，并且参考其他一些先进的语言，开发出分布性好、安全性高，适合网络开发环境的语言，研究小组的领导人 James Gosling 以他在 Sun 公司办公室外的一棵橡树(oak)将其命名为 Oak，后来才发现已有一种称为 Oak 的计算机语言。由于研发小组成员经常在公司附近的一家咖啡厅喝咖啡，因此最终将咖啡原产地 Java(爪哇)作为新语言的名称，寄托了设计者们浓浓的情意，还把它形象化——在 Java 开发环境中，人们常会见到一杯冒着热气的咖啡图标。

但是，由于商业上的种种原因，当时这些电子产品没有推向市场，Java 也差点夭折。1993 年，Internet 由字符界面发展到图形界面，这加快了 Internet 的发展。1994 年，Sun 公司的元老 Bill Joe 参加了 Green 小组并决定将 Java 用在 Internet 的 WWW 开发中并且取得了设计上的成功，通过互联网让世界无数程序设计人员免费下载使用，从而使 Java 被越来越多的人所认识，也使其越来越受到重视。1995 年，Sun 公司正式推出了 Java 的测试版以及用 Java 开发的浏览器 HotJava，并很快被著名杂志《PC Magazine》、《Time》列入优秀科技产品榜。此后，Netscape、Macromedia、IBM 和 Microsoft 等公司相继宣布支持 Java，该语言从此进入飞速发展的时期。

2. Java 语言的特点

Java 以其跨平台、面向对象、多线程、稳定性好、安全性强、兼具编译型语言和解释型语言特点，以及能与 Internet 完美结合等特征而取得成功。Java 的特点有以下几个方面。

(1) Java 语言最突出的特点是跨平台性，也叫与平台无关性。就是说，用 Java 语言编写的程序可以在任何一台计算机上运行，这是因为 Java 语言中没有任何功能与工作平台相关。

(2) Java 语言的第二个重要特点是面向对象。这是指把程序实现的每一个具体功能都作为类，然后由类来构成对象，这种方法会在以后的章节中详细介绍。

(3) Java 语言的第三个特点是多线程。线程是指正在运行的程序，但线程有别于进程，即多个线程共用一个内存区域，也共享同一组系统资源，对每个线程来讲，只有堆栈和寄

存器数据是独立的，所以线程之间进行通信和切换时，系统开销要比进程机制小得多。

(4) Java 语言的第四个特点是具有编译型语言和解释型语言的优点。编译是指一次性把源程序翻译成可以运行的目标程序，以后翻译好的目标程序可以作为一个独立的文件无数次地运行。编译过程所需要的存储空间大，同时，编译所需要的时间较长，但程序执行速度快，C、FORTRAN、Pascal 等都属于编译型语言。解释是指对源程序翻译一句执行一句，翻译和运行交叉进行，如果要运行一次，那就必须重新翻译、执行，翻译完即执行完。这类语言最典型的例子是 BASIC，由于边解释边执行，所以解释型语言的速度远远低于编译型语言。Java 语言是半编译半解释的语言，一个 Java 语言源程序要运行，必须先由 Java 编译器编译成字节码，但这个编译过程是不彻底的，因为字节码不是最终的执行程序，不能在具体的平台上运行，必须由运行系统上的解释器将其翻译成机器语言，Java 解释器采取边解释边执行的方式，但速度仍相当快。

(5) Java 语言还有一个非常重要的特点，就是 Applet 功能以及与此相关的图形功能。Applet 是 Java 特有的一种小应用程序，Java 系统提供特殊的技术，可以使这些小应用程序的功能方便地加入到 Internet 的网页上去，从而为 Internet 网页增加各种图形效果和多媒体功能。

除了上述特点外，Java 还具有稳定性好、安全性高和编程简单等优点。

3. 与 C/C++的区别

熟悉 C 语言和 C++语言的读者一定想搞清这个问题，实际上，Java 确实继承了 C 语言和 C++语言许多优秀的部分，比如 Java 在变量声明、运算符形式、参数传递和流控制等方面和 C 语言、C++语言相同。但是，Java 和 C 语言、C++语言存在许多差别，主要有以下几个方面。

(1) Java 对内存的分配是动态的，它采用面向对象的机制，采用运算符 new 为每个对象分配内存空间，而且实际内存还会随程序的运行情况而改变，同时，Java 能自动回收不再使用的内存，具有自动垃圾搜集功能。

(2) Java 不使用 goto 语句，而用 try-catch-finally 异常处理语句来代替 goto 语句处理除错的功能。

(3) Java 不在所有类之外定义全局变量，而是在某个类中定义一种公用静态的变量来完成全局变量的功能。

(4) Java 不支持头文件。

(5) Java 不支持宏定义，而是用关键字 final 来定义常量。

(6) Java 为每种数据类型都分配固定长度，例如在 Java 中，int 类型总是 32 位的，而 C 语言和 C++语言中对于不同的平台为同一个数据类型分配不同的字节数，例如同是 int，在 PC 机中为 16 位，而在 VAX-11 中为 32 位，这就造成 C 语言不可移植性，而 Java 则具有跨平台性。

(7) Java 不使用指针，保证了系统的安全性。

4. Java 的应用

由于 Java 语言的众多特点，使它有着很好的应用前景，主要包括以下几方面。

(1) Java 语言由于具有跨平台的特点，使它能很好地用于不同机型、不同操作系统之间

的数据交换和通信，完成协调控制、综合管理等功能。

(2) 用于可视化图形软件和动画软件的设计。Java 语言由于可以设计质量很高的活动图形软件，因此，它对计算机图形学、多媒体通信能提供良好的支持。

(3) 用于计算机交互软件的设计和开发。由于 Java 具有良好的图形功能、可视化及可操作化等优点，为交互软件的设计带来方便。

(4) 为 Internet 网络用户提供生动活泼的带动画的主页。由于 Java 具有 Applet 功能，使其能非常方便地将动画和各种信息嵌入网页，因此，Java 对网络用户具有强大的吸引力。

在经历了以大型机为代表的集中计算模式和以 PC 为代表的分散计算模式之后，互联网的出现使得计算模式进入了网络计算时代。网络计算模式的一个特点是计算机是异构的，即计算机的类型和操作系统不一样，例如 Sun 工作站的硬件是 SPARC 体系，软件是 UNIX 中的 Solaris 操作系统，而 PC 的硬件是 Intel 体系，操作系统是 Windows 或者是 Linux，因此相应的编程语言基本上只适用于单机系统，例如 COBOL、FORTRAN、C、C++等；网络计算模式的另一个特点是代码可以通过网络在各种计算机上进行迁移，这就迫切需要一种跨平台的编程语言，使得用它编写的程序能够在网络中的各种计算机上正常运行，Java 语言就是在这种需求下应运而生的。正是因为 Java 语言符合了互联网时代的发展要求，才使它获得了巨大的成功。

Java 语言是和 Internet 同步发展起来的一种新型网络语言，是近二十年计算机软件环境中的最有意义的进步之一。Java 语言在网络中的地位同超文本链接标记语言(HypertextMarkup Language，HTML)一样重要。Java 语言是一种强有力的网络编程语言，它最大限度地利用了网络资源，Applet 小程序可以跨平台、跨操作系统、跨网络地运行。由于 Applet 代码小，因而易于在网络上快速地下载和发送，且具有不需要修改应用程序就可以增加 Web 页的新功能。Java 还为编程技术人员提供了许多公用的系统接口。随着 Internet 在全世界范围内的广泛流行，以及在各个领域的渗透，Java 语言已被各行各业的人士所接受。

1.2　Java 的工作原理

要了解 Java 的工作原理，需要了解 Java 虚拟机、Java 虚拟机体系结构和代码安全性检查机制，下面分别进行介绍。

1.2.1　Java 虚拟机

Java 虚拟机是软件模拟的计算机，可以在任何处理器上(无论是在计算机中还是在其他电子设备中)安全并且兼容地执行保存在.class 文件中的字节码。Java 虚拟机的“机器码”保存在.class 文件中，有时也称之为字节码文件。Java 程序的跨平台主要是指字节码文件可以在任何具有 Java 虚拟机的计算机或者电子设备上运行，Java 虚拟机中的 Java 解释器负责将字节码文件解释成为特定的机器码来运行。Java 源程序需要通过编译器编译成为.class 文件(字节码文件)，Java 程序的编译和执行过程如图 1-1 所示。

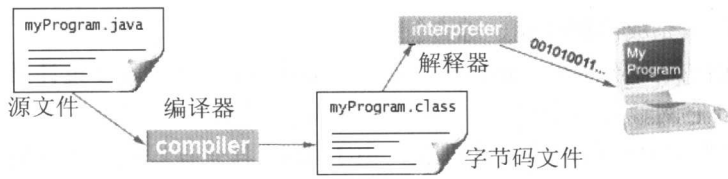


图 1-1 编译和执行过程

但是，Java 虚拟机的建立需要针对不同的软硬件平台做专门的实现，既要考虑处理器的型号，也要考虑操作系统的种类。如图 1-2 所示，目前在 SPARC 结构、X86 结构、MIPS 和 PPC 等嵌入式处理芯片上，在 UNIX、Linux、Windows 和部分实时操作系统上都有 Java 虚拟机的实现。

图 1-2 Java 虚拟机针对不同软硬件平台的实现

1.2.2 Java 虚拟机体系结构

Java 虚拟机由 5 个部分组成：一组指令集、一组寄存器、一个栈、一个垃圾回收堆(Garbage-collected-heap)和一个方法区域。这 5 部分是 Java 虚拟机的逻辑成分，不依赖任何实现技术或组织方式，但它们的功能必须在真实机器上以某种方式实现。

1. Java 指令集

Java 虚拟机支持 248 个字节码。每个字节码代表一种基本的 CPU 运算，例如，把一个整数加到寄存器、子程序转移等。Java 指令集相当于 Java 程序的汇编语言。Java 指令集中的指令包含一个单字节的操作符，用于指定要执行的操作，还有 0 个或多个操作数，提供操作所需的参数或数据。虚拟机的内层循环的执行过程如下：

```
do {
    取一个操作符字节;
    根据操作符的值执行一个动作;
}while(程序未结束)
```

由于指令系统的简单性，使得虚拟机执行的过程十分简单，从而有利于提高执行的效率。指令中操作数的数量和大小是由操作符决定的。如果操作数比一个字节大，那么它存储的顺序是高位字节优先。例如，一个 16 位的参数存放时占用两个字节，其值为：第一个字节×256+第二个字节。字节码指令流一般只是字节对齐的。指令 tableswitch 和 lookup 是例外，在这两条指令内部要求强制的 4 字节边界对齐。

2. 寄存器

Java 虚拟机的寄存器用于保存机器的运行状态，与微处理器中的某些专用寄存器类似。Java 虚拟机的寄存器有 4 种。

- pc：Java 程序计数器。
- optop：指向操作数栈顶端的指针。
- frame：指向当前执行方法的执行环境的指针。
- vars：指向当前执行方法的局部变量区第一个变量的指针。

Java 虚拟机是栈式的，它不定义或不使用寄存器来传递或接受参数，其目的是为了保证指令集的简洁性和实现时的高效性(特别是对于寄存器数目不多的处理器)。所有寄存器都是 32 位的。

3. 栈

Java 虚拟机的栈有 3 个区域：局部变量区、运行环境区和操作数栈区。

(1) 局部变量区

每个 Java 方法使用一个固定大小的局部变量集，按照与 vars 寄存器的字偏移量来寻址。局部变量都是 32 位的。长整数和双精度浮点数占据了两个局部变量的空间，却按照第一个局部变量的索引来寻址。例如，一个具有索引 n 的局部变量，如果是一个双精度浮点数，那么它实际占据了索引 n 和 n+1 所代表的存储空间。虚拟机提供了把局部变量中的值装载到操作数栈的指令，也提供了把操作数栈中的值写入局部变量的指令。

(2) 运行环境区

在运行环境中包含的信息用于动态链接、正常的方法返回以及异常传播。

- 动态链接。运行环境包括指向当前类和当前方法的解释器符号表的指针，用于支持方法代码的动态链接。动态链接把符号形式的方法调用翻译成实际方法调用，装载必要的类以解释还没有定义的符号，并把变量访问翻译成与这些变量运行时的存储结构相应的偏移地址。动态链接方法和变量使得方法中使用的其他类的变化不会影响到本程序的代码。
- 正常的方法返回。如果当前方法正常结束，在执行了一条具有正确类型的返回指令时，调用的方法会得到一个返回值。执行环境在正常返回的情况下恢复调用者的寄存器，并把调用者的程序计数器增加一个恰当的数值，以跳过已执行过的方法调用指令，然后在调用者的执行环境中继续执行下去。
- 异常传播。异常情况在 Java 中被称作 Error(错误)或 Exception(异常)，是 Throwable 类的子类，引起异常的原因有两个：一是动态链接错误，如无法找到所需的 class 文件；二是运行时错误，如对一个空指针的引用等。

当异常发生时，Java 虚拟机采取如下措施。

① 检查与当前方法相联系的 catch 子句表。每个 catch 子句都包含其有效指令范围、能够处理的异常类型，以及处理异常的代码块地址。

② 与异常相匹配的 catch 子句应该符合下列条件：造成异常的指令在其指令范围之内，发生的异常类型是其能处理的异常类型的子类型。如果找到了匹配的 catch 子句，那么系统转移到指定的异常处理块执行；如果没有找到异常处理块，则重复寻找匹配的 catch 子句

的过程，直到当前方法的所有嵌套的 catch 子句都被检查过。

③ 由于虚拟机从第一个匹配的 catch 子句处继续执行，所以 catch 子句表中的顺序很重要。因为 Java 代码是结构化的，因此总可以把某个方法的所有的异常处理器都按序排列到一个表中，对任意可能的程序计数器的值，都可以用线性的顺序找到合适的异常处理块，以处理在该程序计数器值下发生的异常情况。

④ 如果找不到匹配的 catch 子句，那么当前方法得到一个“未截获异常”的结果并返回到当前方法的调用者，好像异常刚刚在其调用者中发生一样。如果在调用者中仍然没有找到相应的异常处理块，“未截获异常”的信息将被继续传播下去。如若被传播到最顶层，那么系统将调用一个默认的异常处理块。

(3) 操作数栈区

机器指令只从操作数栈中取操作数，对它们进行操作，并把结果返回到栈中。选择栈结构的原因是：在只有少量寄存器或非通用寄存器的机器(如 Intel 486)上，也能够高效地模拟虚拟机的行为。操作数栈是 32 位的。它用于给方法传递参数，并从方法接收结果，也用于支持操作的参数，并保存操作的结果。例如，iadd 指令将两个整数相加。相加的两个整数应该是操作数栈顶的两个字。这两个字是由先前的指令压进堆栈的。这两个整数将从堆栈弹出、相加，并把结果压回到操作数栈中。

4. 无用单元收集堆

Java 的堆是一个运行时的数据区，类的实例(对象)从中分配空间。Java 语言具有垃圾回收能力：程序员不用自己编写代码来释放内存。Java 不规定具体使用的垃圾回收算法，可以根据系统的需求使用各种算法。

5. 方法区

方法区与传统语言中的编译后代码或是 UNIX 进程中的正文段类似。方法区保存方法代码(编译后的 Java 代码)和符号表。在当前的 Java 实现中，方法代码不包括在垃圾回收堆中，但计划在将来的版本中实现。类文件是 Java 语言的执行代码文件。为了保证类文件的平台无关性，Java 虚拟机规范中对类文件的格式也作了详细的说明。其具体细节参考 Sun 公司的 Java 虚拟机规范。

1.2.3 代码安全性检查机制

安全和方便总是相对矛盾的。Java 编程语言的出现使得客户端机器可以方便地从网络上下载 Java 程序到本机上运行，但是如何保证该 Java 程序不携带病毒或者不怀有其他险恶目的呢？如果 Java 语言不能保证执行的安全性，那么它就不可能存活到今天。虽然有时候少数程序员会抱怨说 Applet 连文件系统也不能访问，但是正是各种安全措施的实行才确保了 Java 语言的生存。

字节码的执行需要经过 3 个步骤，首先由类装载器(class loader)负责把类文件(.class 文件)加载到 Java 虚拟机中，在此过程需要检验该类文件是否符合类文件规范；其次字节码校验器(bytecode verifier)检查该类文件的代码中是否存在着某些非法操作，例如 Applet 程序

中写本机文件系统的操作；如果字节码校验器检验通过，由 Java 解释器负责把该类文件解释成为机器码进行执行。Java 虚拟机采用的是“砂箱”运行模式，即把 Java 程序的代码和数据都限制在一定内存空间里执行，不允许程序访问该内存空间外的内存，如果是 Applet 程序，还不允许访问客户端机器的文件系统。

1.3　Java 平台

完整的 Java 体系结构包括 4 个组件：Java 编程语言、Java 类文件格式、Java 应用程序编程接口(Application Programming Interface，API)和 Java 虚拟机。Java 虚拟机与基本 API 构成了 Java 平台，也称为 Java 运行时环境(Java Runtime Environment，JRE)，它位于操作系统之上。

1.3.1　Java 常用包

Java 应用程序编程接口分为 3 大平台。

(1) Java 2 Platform，Standard Edition(Java 2 平台，标准版)，简称 J2SE。这个平台包含 Java 核心类和 GUI 类。

(2) Java 2 Platform，Enterprise Edition(Java 2 平台，企业版)，简称 J2EE。这个平台包含开发基于 Web 的应用程序的类和接口，如 Servlet、Java Server Pages 以及 Enterprise JavaBeans 等。

(3) Java 2 Platform，Micro Edition(Java 2 平台，微观版)，简称 J2ME。这个平台对传呼机、移动电话、掌上电脑、汽车导航系统或其他无线设备等产品提供优化的运行时环境。

本书介绍和使用的是 J2SE，它是学习和开发 Java 的最佳起点。几年来，Sun 公司已经推出了许多不同的 JDK(Java Development Kit，Java 开发工具包)版本。目前的 JDK 版本有 JDK 1.0、JDK 1.1、JDK 1.2、JDK 1.4 和 JDK 1.5。而从 JDK 1.2 以后的版本简称 Java 2。

读者可以随时查看 Sun 公司的 Java 网站，找出可下载的最新版本。通常在 12~18 个月的时间周期内 Sun 公司就会推出一个新的版本，每次更新都会包括质量和性能方面的改进，也会增加少量新的特征。

Java 提供了强大的应用程序接口，即 Java 类库。它包括大量已经设计好的工具类，帮助程序员进行字符串处理、绘图、数学计算和网络应用等方面的工作。下面简单介绍 Java 核心类库中常用的组件包。

1. java.lang 包

在所有的 Java API 类库中，java.lang 包是最重要的，它提供了 Java 语言的核心类库，包含了运行 Java 程序必不可少的系统类，如基本数据类型、基本数学函数、字符串处理、线程管理和异常处理类等。运行 Java 程序时，系统会自动加载 java.lang 包，即这个包的加载是默认的。

2. java.io 包

java.io 包提供了一系列用来读写文件或其他的输入输出流。其中有基本输入/输出类、缓存流类、比特数组与字符串流类、数据流类、文件流类、管道类、流连接类和异常类等。

3. java.util 包

java.util 包提供了 Java 语言中的一些低级的实用工具，如数据结构类、日期类、随机数类、属性类、观测器类和异常类等。

4. java.awt 包

java.awt 包是 Java 语言用来构建图形用户界面(GUI)的类库，包括许多界面元素和资源。java.awt 包提供 Java 语言中的图形类、组成类、容器类、排列类、几何类、事件类和工具类等。

5. java.net 包

java.net 包包含一些与网络相关的类和接口，以方便应用程序在网络上传输信息，如主机名解析类、实现套接字通信的 Socket 类和 ServerSocket 类、资源定位器(URL)类等。

6. java.applet 包

java.applet 包是用来实现运行于 Internet 浏览器中的 Java Applet 的工具类库。它包含用于产生 Applet 的类和用于 Applet 通信的类。Applet 类称为小应用程序类，通常所说的 Applet 程序必须继承该类。Applet 是一种专门化的面板，需要嵌入到 HTML 网页中，与 Java 语言兼容的浏览器一起执行。

7. java.awt.event 包

java.awt.event 包是对 JDK 1.0 版本中原有的 Event 类的一个扩充，它使得程序可以用不同的方式来处理不同类型的事件，该包中定义了许多不同类型的事件监听器类，使每个图形界面元素本身可以处理它上面的事件。

除了上述的常用 Java 组件包外，Java 类库中还有很多实用的组件包，并且还在不断地扩充，请查看相关 Java 文档。

1.3.2 Java 工具

Java 2 平台标准版带有一些开发工具，可以编译、执行和调试 Java 程序，这些工具以命令行的方式使用。除 appletviewer 和 policytool 外，这些工具都不提供图形用户界面。这里介绍几个与 JVM 有关的工具。关于 Java 实用程序工具的完整清单，参见 Sun 公司的 Java 网站(http://java.sun.com)。

Java 2 SDK 1.5 软件包可以从 http://java.sun.com 免费下载，由于 Java 软件包受不同操作系统的支持，所以下载得到的文件各不相同。对于运行在 Microsoft Windows 平台上的软件包，下载得到的文件为 j2sdk-1_5-win.exe，安装时直接运行该文件即可。安装后可以设置系统的 PATH 和 CLASSPATH 环境变量。

为了在使用 javac 编译 Java 源程序时以及使用 java 或 appletviewer 运行 Java 程序时无需指定其路径，需要编辑 AUTOEXEC.BAT 文件，在 PATH 环境变量中增加 bin 目录的路径，例如：PATH="%PATH%"；d:\jsdk1.5\bin。

为了在工作目录下使用其他目录中的类，可以在 AUTOEXEC.BAT 文件中设置 CLASSPATH 环境变量。设置方法为：

```
set CLASSPATH=path1;path2…
```

如果 CLASSPATH 环境变量被设置成不正确的值，或在启动文件或脚本程序中设置了不正确的路径，则可以通过使用下列命令清除 CLASSPATH：

```
set CLASSPATH=
```

1. Java 编译器

J2SE 自带的 Java 编译器是 javac.exe。Java 源程序文件的扩展名为.java，它是标准的 ASCII 文本文件，利用 javac 工具，可以将其编译为可执行的 Java 字节码文件，即扩展名为.class 的文件。javac 编译器会为每一个类生成一个对应的.class 文件，无论这些类是否在同一个文件中。

(1) javac 命令语法

javac 命令行格式为：

```
javac [options] [sourcefiels] [@files]
```

命令中参数说明如下：

① 参数可按任意次序排列。

② options 指命令行选项。

③ sourcefiles 指一个或多个要编译的源文件。若一次要对多个文件进行编译，文件名之间用空格分隔。例如：

```
javac hello.java helloworld.java
```

④ @files 指一个或多个列表文件。列表文件中包含了要编译的一个或多个源程序文件。若一次要对多个列表文件进行编译，列表文件名之间用空格分隔。例如：

```
javac @f1 @f2
```

(2) javac 命令选项

javac 编译器有一些标准选项，目前的开发环境支持这些标准选项，将来的版本也将支持它。还有一批附加的非标准选项是目前的虚拟机实现所特有的，将来可能要有变化。非标准选项以-X 打头。下面简单介绍常用的一些选项。

① -classpath <path>

设置用户类路径，它将覆盖 CLASSPATH 环境变量中的用户类路径，路径项用分号(;)分隔。若既未指定 CLASSPATH 又未指定-classpath，则用户类路径是当前目录。

② -d <directory>

设置类文件的目标目录。如果某个类是一个包的组成部分，则 javac 应将该类文件放入